CIS 5500 Project Outline: Funding-Aware Market Maker

Fall 2025

Team

- **Names:** Gaurav Malhotra, Albert Opher, Ishaan Shah, Madhav Sharma
- **Emails:** gmal2005@seas.upenn.edu, albert.w.opher.iv@gmail.com, ishaan8@wharton.upenn.edu, madhavsh@wharton.upenn.edu
- **GitHub:** GauravMalhotra010, Albinator3000, IshaanShah25082005, Madhav-Sharma-07

Motivation / Problem Description Perpetual futures charge/credit funding every eight hours. Around these funding timestamps, order flow and positioning often become imbalanced, creating short, predictable drifts and increased adverse selection for liquidity providers (market makers). Our application helps a market maker *decide when and by how much to widen quotes* and adjust inventory targets around funding events by analyzing large historical datasets (minute klines, funding rates, open interest, and optional premium index). The system provides event studies, regime screens, and a rules lab to turn insights into actionable parameters.

Features Definitely implement

1. **Event Study Explorer:** For any symbol and date range, visualize pre/post-funding cumulative returns (CAR), volatility, OI overlays, and funding deciles.
2. **Regime Screener:** Rank symbols/windows where funding absolute value and OI percentile are jointly extreme; export ranked tables.
3. **Quote-Width Rules Lab:** Interactive rule builder (e.g., widen by Δ when funding decile$\geq$ 9 and RV>p75); compute historical hit rate and P&L proxy.
4. **Timing Dashboard:** Show pre-optimization vs post-optimization runtimes and query plans for at least four complex SQL queries.
5. **Dataset Health:** Row counts after cleaning, null/missing diagnostics, and reproducible ingestion logs.

Stretch (time permitting)

- Premium/basis integration (`premiumIndexKlines`) and basis-aware rules.
- Simple multi-symbol portfolio simulator with risk caps.
- Scheduling service to keep RDS in sync with new monthly archives.

Pages (multi-page web app)

Page	Description
Home	Project overview, dataset coverage, and quick links to studies.
Event Study Explorer	Choose symbol and window length; plot CAR in $[-60, +180]$ minutes around each funding print, with OI and funding decile bands.
Regime Screener	Filter by funding$\lvert\cdot\rvert$ decile and OI percentile; returns ranked symbol–event windows with stats and export.
Rules Lab	Define quote-widening / inventory rules; compute historical outcomes (hit rate, average markout) and suggested parameters.
Timings Dashboard	For 4+ complex SQL queries, display EXPLAIN plans and pre/post runtime measurements with notes on indexes/MVs.
Data Admin	Row counts after cleaning (guarantee $>$ 100,000 rows), ingestion status, and backfills.

Datasets, Scale, and Overlap

Dataset	Provider	Granularity / Notes	Access
USDT-Margined Futures **Klines**	Binance	1-minute OHLCV + taker metrics; monthly archives; 50+ symbols. For just 10 symbols $\times$ 12 months we exceed 5,000,000 rows.	https://data.binance.vision/?prefix=data/futures/um/monthly/klines/
Funding Rate History	Binance	8h funding timestamps and rates per symbol; CSV archives + REST.	https://data.binance.vision/?prefix=data/futures/um/monthly/fundingRate/
Open Interest (OI)	Binance	OI snapshots; sufficient cadence around funding windows (via REST, cached to CSV).	https://developers.binance.com/docs/derivatives/usds-margined-futures/market-data/rest-api/Open-Interest-Statistic
Optional: Premium Index Klines	Binance	Basis between perp and spot index; joinable by (symbol, timestamp).	https://developers.binance.com/docs/derivatives/usds-margined-futures/market-data/rest-api/Premium-Index-Kline-Dat

Scale guarantee (post-cleaning): We will ingest at least **12 months** of 1-minute klines for $\geq$ 10 active symbols ($\approx$ 5,256,000 klines), plus funding and OI joins, ensuring far above the

100,000 row threshold even after filters and deduplication.

Relational Schema (ER Diagram)

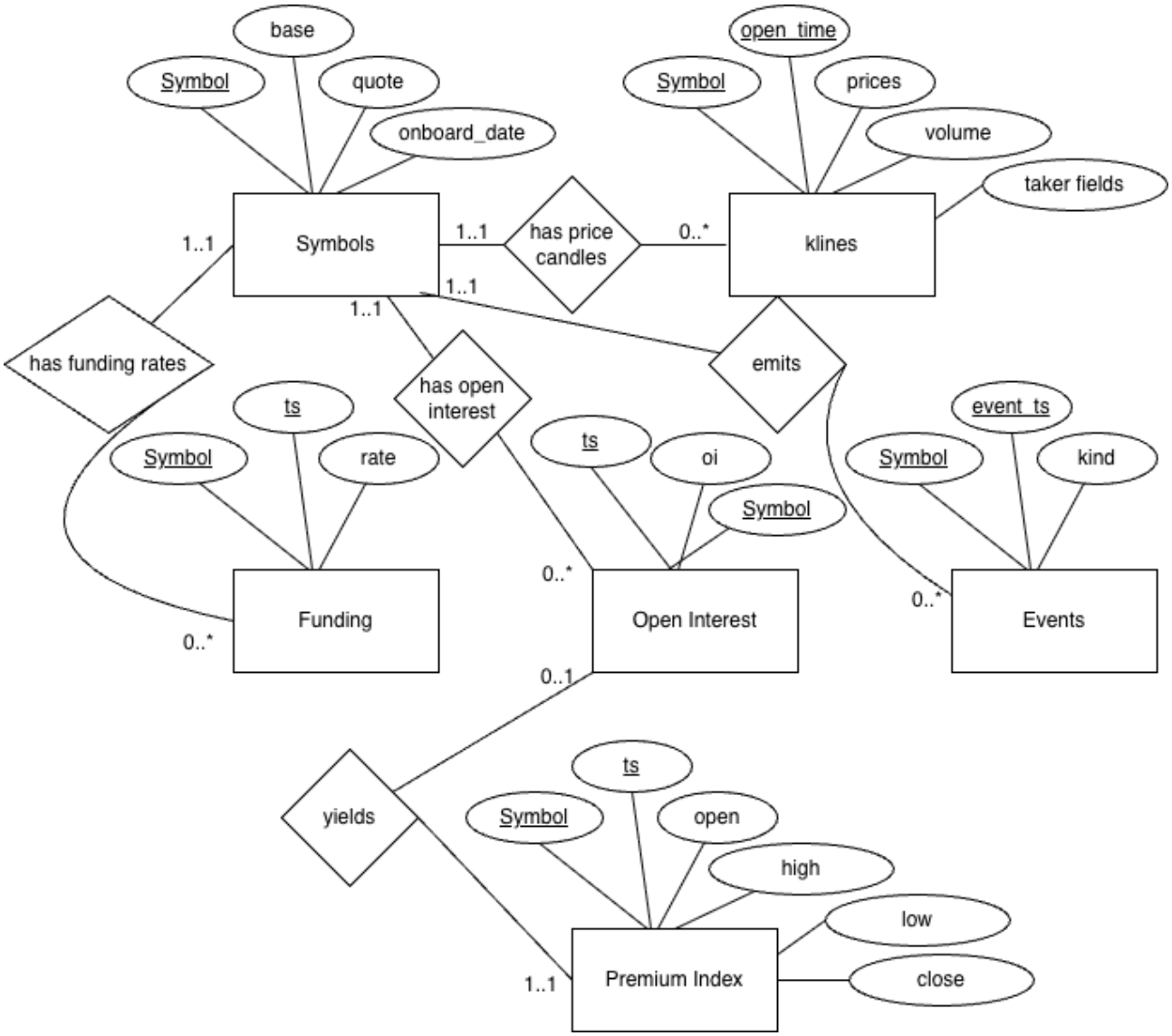

Figure 1: ER Diagram

SQL DDL (MySQL 8.0)

```
-- Schema: funding_aware_mm (MySQL 8.0)
CREATE DATABASE IF NOT EXISTS funding_aware_mm;
USE funding_aware_mm;

CREATE TABLE symbols (
  symbol        VARCHAR(20) PRIMARY KEY,
  base          VARCHAR(16) NOT NULL,
  quote         VARCHAR(16) NOT NULL,
  onboard_date  DATE
) ENGINE=InnoDB;

CREATE TABLE klines (
  symbol           VARCHAR(20) NOT NULL,
  open_time        DATETIME    NOT NULL,
```

```
  close_time       DATETIME     NOT NULL,
  open_price       DECIMAL(18,8) NOT NULL,
  high_price       DECIMAL(18,8) NOT NULL,
  low_price        DECIMAL(18,8) NOT NULL,
  close_price      DECIMAL(18,8) NOT NULL,
  volume_base      DECIMAL(24,8) NOT NULL,
  taker_buy_base   DECIMAL(24,8) DEFAULT 0,
  taker_buy_quote  DECIMAL(24,8) DEFAULT 0,
  n_trades         INT          DEFAULT 0,
  PRIMARY KEY (symbol, open_time),
  CONSTRAINT fk_klines_symbol FOREIGN KEY (symbol) REFERENCES symbols(symbol)
) ENGINE=InnoDB;

-- Optional monthly partitioning (MySQL RANGE on YYYYMM integer)
ALTER TABLE klines
  ADD COLUMN open_time_ym INT AS (YEAR(open_time)*100 + MONTH(open_time)) STORED,
  ADD INDEX idx_klines_sym_time (symbol, open_time),
  ADD INDEX idx_klines_ym (open_time_ym),
  PARTITION BY RANGE (open_time_ym) (
    PARTITION p202401 VALUES LESS THAN (202402),
    PARTITION p202402 VALUES LESS THAN (202403)
    -- ... add via ALTER as you load more months
  );

CREATE TABLE funding (
  symbol   VARCHAR(20) NOT NULL,
  ts       DATETIME    NOT NULL,
  rate     DECIMAL(12,8) NOT NULL,
  PRIMARY KEY (symbol, ts),
  CONSTRAINT fk_funding_symbol FOREIGN KEY (symbol) REFERENCES symbols(symbol)
) ENGINE=InnoDB;

CREATE TABLE open_interest (
  symbol   VARCHAR(20) NOT NULL,
  ts       DATETIME    NOT NULL,
  oi       DECIMAL(24,8) NOT NULL,
  PRIMARY KEY (symbol, ts),
  CONSTRAINT fk_oi_symbol FOREIGN KEY (symbol) REFERENCES symbols(symbol)
) ENGINE=InnoDB;

CREATE TABLE premium_index (
  symbol   VARCHAR(20) NOT NULL,
  ts       DATETIME    NOT NULL,
  open_val DECIMAL(18,10) NOT NULL,
  high_val DECIMAL(18,10) NOT NULL,
  low_val  DECIMAL(18,10) NOT NULL,
  close_val DECIMAL(18,10) NOT NULL,
  PRIMARY KEY (symbol, ts),
  CONSTRAINT fk_prem_symbol FOREIGN KEY (symbol) REFERENCES symbols(symbol)
) ENGINE=InnoDB;

CREATE TABLE events (
  symbol    VARCHAR(20) NOT NULL,
  event_ts  DATETIME    NOT NULL,
  kind      ENUM('funding','pseudo') NOT NULL,
  PRIMARY KEY (symbol, event_ts),
  CONSTRAINT fk_events_symbol FOREIGN KEY (symbol) REFERENCES symbols(symbol)
) ENGINE=InnoDB;
```

```
-- Views / derived features (MySQL 8.0 supports windows)
CREATE OR REPLACE VIEW minute_returns AS
SELECT symbol,
       open_time AS ts,
       LOG(close_price) - LOG(LAG(close_price) OVER (PARTITION BY symbol ORDER BY op
       STDDEV_SAMP(LOG(close_price) - LOG(LAG(close_price) OVER (PARTITION BY symbol
         OVER (PARTITION BY symbol ORDER BY open_time ROWS BETWEEN 30 PRECEDING AND
FROM klines;

-- Helper: expanded funding windows (per minute around each funding ts)
-- (materialize in app via ETL to avoid heavy on-the-fly generation)
```

Data Cleaning & Pre-processing Objectives: (i) guarantee ¿100k clean rows; (ii) exact symbol/time alignment; (iii) reproducibility.

Steps

1. **Acquisition:** Download monthly 1-minute kline ZIPs for ≥ 10 symbols and CSV funding archives; fetch OI via REST in chunks, cache to CSV.
2. **Validation:** Check column types, monotonic timestamps, non-negative prices/volumes; drop obvious outliers (e.g., zero/NaN prices).
3. **Normalization:** UTC unify timestamps to minutes; standardize symbol case; enforce deduplication on (symbol, open_time) and (symbol, ts).
4. **Integrity:** Populate `symbols` from kline symbol list; enforce FK constraints by removing stray records.
5. **Window labels:** Build `events` from `funding` (and optional pseudo events) and pre-compute per-minute expansions in an ETL step.
6. **Load:** Use MySQL `LOAD DATA LOCAL INFILE` or `mysqlimport`; bulk insert in sorted order by (symbol, time) to maximize index locality; then add partitions for future months.

Technologies

- **Database:** MySQL 8.0 on AWS RDS. InnoDB, partitions, window functions, views.
- **Backend:** Python (FastAPI or Flask) with SQLAlchemy/MySQL Connector/Python.
- **ETL:** Python (pandas, pyarrow), gzip; optional S3 staging.
- **Frontend:** React + Vite; plots via Plotly.
- **Infra:** AWS RDS (db.t3.small or similar), EC2 or Render for API, GitHub Actions for CI lint/tests.

Initial Setup Completed

- Repository initialized (private), base Python/Node scaffolds, `.env` template.
- RDS parameter group prepared; inbound security group restricted to team IPs.
- Prototype loader that ingests one symbol for one month and verifies FK/indexes.

Division of Responsibilities

- **Gaurav:** ETL pipelines, ingestion validation, and partition maintenance.
- **Albert:** Backend API (queries, pagination) and timings dashboard.
- **Ishaan:** Data modeling, complex SQL design/optimization, and event study math.
- **Madhav:** Frontend UI/UX (Explorer, Screener, Rules Lab) and integration tests.

Initial Query Set (10 total; ≥4 complex)

1. **Event CAR (*complex*).** For each funding event, compute CAR over $[-60, +180]$ minutes relative to market beta; return by symbol and event date.
2. **Decile Monotonicity (*complex*).** Bucket events by daily funding-rate deciles; compute mean 60-minute markout per decile and test monotonic trend.
3. **Funding × OI Regimes (*complex*).** Windows where $|rate| > p90$ and OI percentile$> p75$; rank by mean post-event return.
4. **Universal Safety (*complex*).** Symbols that never have negative 30-minute CAR after funding when realized vol<median (`NOT EXISTS`).
5. Hour-of-Day Seasonality: Average CAR by funding hour vs. pseudo events.
6. Basis Interaction: Join premium index; is large positive basis associated with larger adverse selection?
7. Volatility Conditioning: Condition CAR on pre-event rv_30m buckets.
8. Look-Ahead Bias Check: Ensure only information available up to funding time is used; list any violations.
9. Cross-Symbol Panel: Sector/group aggregates (e.g., majors vs alts) with de-meaning.
10. Rule Outcomes: For a ruleset (Δ, thresholds), compute hit rate and average markout.

Optimization Plan

- **Indexes:** Covering indexes on (symbol, time) in all fact tables; secondary index on generated month key (`open_time_ym`).
- **Partitioning:** Monthly partitions on `klines`, weekly on `funding/open_interest`; prune scans during event windows.
- **Materialized Tables:** Pre-expanded funding windows and decile labels stored as physical tables for repeatable interactive queries.
- **Rewrite:** Flatten nested subqueries into CTEs; pre-aggregate per minute before joining to funding events; ensure sorted merge joins.
- **Explain/Analyze:** Capture plans and wall-clock before/after; target $>$ 15s naive $\rightarrow <$ 1s optimized for at least 4 complex queries.

Compliance with Feedback

- **Row count after cleaning:** Guaranteed $>$ 100,000 rows via 12 months of 1m klines for $\geq$ 10 symbols (millions of rows expected).
- **Complex queries:** At least **four** clearly marked complex queries (items 1–4) with optimization evidence on the Timings Dashboard.